Shelf Manager
User Manual

Revision history:

- 26.11.2014 REV 1.6: Changed lanconfig command, added support for two separate IPMBs in CPCI
  added “hostname” and “shelfaddr” commands
- 13.05.2014 REV 1.5: Added IPMB Interface chapter
- 13.05.2014 REV 1.4: Added lanconfig command
- 11.11.2013 REV 1.3: AES added to SNMP

WEB section content updated

- 04.02.2013 REV 1.0: - First Draft

Table of Contents

1 Product description and functions....4
1.1 Voltage monitoring....4
1.2 Temperature monitoring....4
1.3 Fan monitoring and control....4
1.3.1 Fan control....5
1.4 Digital Inputs....6
1.5 Digital Outputs....6
2 Sensor Data Records(SDRs)....7
2.1 Accessing Local Sensor Information....7
2.2 Local Sensor Numbers....8
2.3 Sensor Event Log (SEL)....10
3 IPMB interface....11
3.1 Supported command list....11
3.2 Shutting down a CPCI system....13
4 Ethernet interface....15
4.1 WEB....15
4.1.1 Overview....15
4.1.2 Architecture....17
4.1.3 REST Commands....17
4.1.4 How it all works....23
4.2 Telnet....24
4.3 SNMP....25
4.4 RMCP....26
5 RS232 serial interface....26
6 Command Line Interface (CLI)....26
6.1 voltage command....27
6.2 fan command....28
6.3 temp command....28
6.4 input command....29
6.5 output command....29
6.6 local_sensor command....30
6.6.1 Changing a threshold's value | Disabling a Threshold....33
6.6.2 Changing the hysteresis....33

6.6.3 Changing the outputs linked to a sensor event....33
6.6.4 Changing the activelevel....34
6.6.5 Changing the fancontrol mask....34
6.6.6 Changing the output type....34
6.6.7 Changing the input type....35
6.6.8 Asserting/Deasserting an output....35
6.6.9 Changing the user's permission to assert/deassert an output....35
6.6.10 Changing the timeout for an input....35
6.6.11 Changing the debounce option....36
6.7 fancontrol command....36
6.7.1 Viewing the fancontrol status....37
6.7.2 Setting a fangroup for auto control....37
6.7.3 Viewing the temperature sensors associated with each fan group....37
6.7.4 Setting a fangroup for manual control....37
6.7.5 Changing the manual fancontrol level....37
6.7.6 Changing the minimum level parameter of the fancontrol algorithm....38
6.7.7 Changing the temp0,temp1,temp2 parameters of the fancontrol algorithm....38
6.8 pwm command....38
6.9 xmodem command....39
6.10 logout command....39
6.11 scispeed command....39
6.12 passw command....39
6.13 reboot command....39
6.14 uptime command....40
6.15 saveenv command....40
6.16 version command....40
6.17 restore command....40
6.18 lanconfig - command....40
6.19 sol command....41
6.20 date command....41
6.21 time command....41
6.22 sel command....42
6.23 help command....42
6.24 snmp command....42
6.25 eth command....42
6.26 tftp command....43
6.27 shelfaddr command....43
6.28 hostname command....43
7 Restore to factory defaults procedure....43

List of Figures

Figure 1: Fan Control Algorithm....5
Figure 2: Board Shutdown IPMI Protocol....14
Figure 3: Web Interface overview....16
Figure 4: Default Web Page Functional Diagram....22

List of Tables

Table 1: Fan Control Algorithm....5
Table 2: Local Sensor Numbers....9
Table 3: Threshold_Code....29

Table 4: Hysteresis_Code.....29
Table 5: Hex_Outputs Bit Mask.....30
Table 6: Fancontrol_Mask.....30

1 Product description and functions

Samway's Shelf Manager (ShMC) watches over the basic health of the System Platform, reports anomalies, and takes corrective action when needed. The Shelf Manager can retrieve inventory information and sensor readings. The Shelf Manager can also perform basic recovery operations such as power cycle or reset of managed entities. The Shelf Manager provides remote access to the monitored information.

The monitored parameters of the System Platform are measured, or read in, and analyzed at regular intervals. If a parameter value exceeds or falls below a user-defined limit, the ShMC detects this as an "Event". In principle, fixed predefined actions are carried out when the corresponding events occur. These actions are described later. It is also possible to use such Events to trigger one or more of the digital Outputs.

The ShMC uses Sensor Data Records, compliant to IPMI 2.0, to describe the monitored System parameters. For any measured parameter up to 6 thresholds can be defined: lower non critical, lower critical, lower non-recoverable, upper non critical, upper critical and upper non-recoverable. The measured values are retrievable at any time via the RS232 serial interface and via Telnet. In addition, limits and system parameters can be changed at any time with the unit in service. As a result, the ShMC – and hence the connected system - can be controlled and monitored online via any computer with an Internet connection.

To complete the information spectrum, the integrated web page, which the user can design himself, is also continuously updated with the measured values.

1.1 Voltage monitoring

- Monitoring of up to 8 voltages (+3.3V,+5V, +12V, -12V and 4 user defined voltages).
- Up to 6 thresholds can be configured individually for each voltage
- Any limit infringement is an internal event, and can control any of the 16 digital Outputs.
- Voltage monitoring parameters can be changed via the CLI or by upgrading the Sensor Data Records (SDRs) .
- CLI Commands for the voltage monitoring: ***voltage , local_sensor***

1.2 Temperature monitoring

- Monitoring of up to 6 analog temperature sensors (10 Kohm NTC thermistors with ß=3950)
- Up to 6 thresholds can be individually configured for each temperature sensor.
- Temperature range from –20°C to +100°C
- Temperature measurement accuracy +/-3°C (max.)
- Any limit infringement is an internal event, and can control any of the 16 digital Outputs.
- Any temperature sensor can be used in the fan control algorithm(The user can choose for each temperature sensor the fan control groups the sensor is active for.)
- Commands for the temperature monitoring: ***temp , local_sensor***

1.3 Fan monitoring and control

- Monitoring of up to 12 fans.
- Control of PWM fans
- Up to 6 thresholds can be individually configured for each fans speed.
- Speed control via 3 fan control groups
- Any limit infringement is an internal event, and can control any of the 16 digital Outputs.
- Commands for the fan monitoring: ***fan, local_sensor***

1.3.1 Fan control

The Speed of the fans can be controlled using one of the 3 fan control groups. Each group controls an independent PWM[1] signal using a user-defined temperature-speed characteristic.

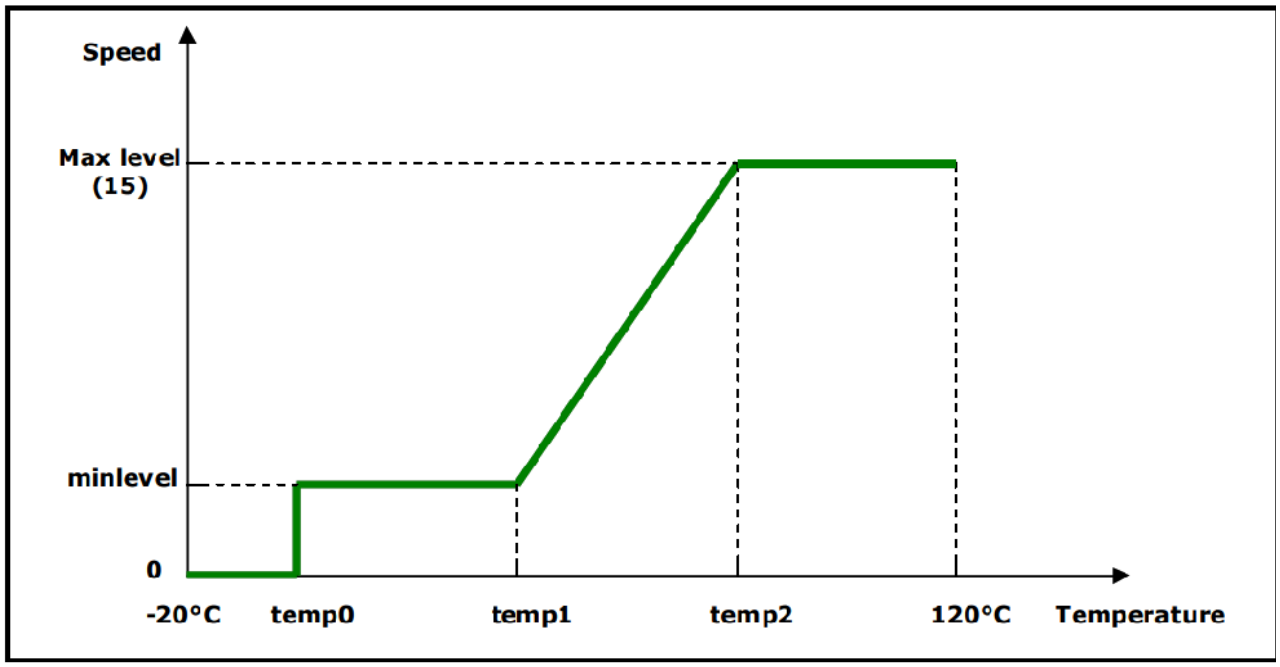

Figure 1: Fan Control Algorithm

The user can define the temperature-speed characteristic using 4 parameters: min level (the minimum level at which the fans operate) and 3 temperature thresholds (temp0, temp1, temp2).

The speed of the fans is split in 15 equal levels. At level 0 the fans are stopped and at level 15 the fans are running full speed.

The 3 temperature thresholds split the operating range of the fan in 4 control regions:

Temperature T	**Fans Behavior:**
T< temp0	Stopped
temp0 < **T** < temp1	Running at min level
temp1<**T**<temp2	Running at a speed level proportional to **T**
temp2<**T**	Running at full speed

Table 1: Fan Control Algorithm

The Fan control algorithm can use any of the installed temperature sensors. The user can choose the active temperature sensor for each fan control group. The temperature that drives the algorithm is the maximum value of all the temperature sensors active for the respective fan control group. If no temperature sensors are active for a particular fan group, the fan control algorithm sets

1 The ShMC can output 3 PWM signals with frequencies between 1KHz and 125KHz. The frequency of PWM1 is equal to that of PWM2, but the duty factors of the 2 PWM signals are independent.

the fans to run at the maximum level.

The 3 fan control groups are independent, each have their own minlevel, temp0, temp1, temp2 parameters and each control a different PWM signal.

The automatic fan control algorithm can also be disabled. In this case the fan speed is set at a fixed manual level. Each fan control group has it's on own manual fan level which can be changed using the CLI via RS232 or Telnet.

Commands useful for fan control:***fancontrol, pwm, temp***

1.4 Digital Inputs

- 16 digital inputs
- Logic level: 5V TTL.
- Each individual input can be declared as active low or active high
- Any active input generates an internal event, and can control any of the 16 digital outputs
- Commands for inputs: ***input, local_sensor***

1.5 Digital Outputs

- 16 digital outputs
- Each individual output can be declared as active low or active high
- Manual setting and clearing of outputs
- Each output can be driven by more than one source. Potential output drivers are:
  - Internal Events : threshold violations for threshold sensors, active levels for discrete sensors
  - External Events: active inputs
- Each output can perform an **AND** or an **OR** function on the driving signals:
  - **AND** outputs are active if all the drivers are active
  - **OR** outputs are active when at least one of the drivers is active
- Commands for outputs: ***output, local_sensor***

2 Sensor Data Records(SDRs)

The ShMC uses Sensor Data Records(SDRs), compliant to IPMI 2.0, to describe the monitored System parameters. SDRs are records that contain information about the type of sensors in the platform, sensor threshold support and event generation capabilities

Using SDRs the user can:

- specify up to **6** thresholds
- decide the way the software treats a limit infringement(events and outputs can be enabled or disabled for each individual threshold)
- **set a name for every sensor**
- change the way the ShMC operates by changing the sensor definitions
- enable/disable the monitoring of a sensor by loading/removing its SDR
- change monitoring parameters by CLI commands or by changing the SDRs

For more information on SDRs you could refer to - *IPMI - Intelligent Platform Management Interface Specification Second Generation v2.0.*

2.1 Accessing Local Sensor Information

The ShMC monitors many parameters of the system platform. Every monitored parameter has one, and only one, software sensor attached. So, each sensor corresponds to only one monitored parameter.

For accessing local sensor information there are two options:

- displaying information for multiple sensors
- accessing each sensor individually using its number

For accessing all the sensors the command ***local_sensor*** is used. Also the user can access information regarding sensors of a particular type using one of the commands: ***voltage, current, temp, fan, input, output.***

All the commands that display sensor information use the same header:

```
----------------------------Sensor List-----------------------------------
--no--Name-------------Type--Value--Unit---State--------------------------
```

The header displays:

- **no** → sensor number; a unique number that identifies a particular sensor.
- **Name** → sensor's name
- **Type** → sensor type : threshold or discrete
- **Value** → current value
- **Unit** → measuring unit
- **State** → current state for threshold sensors. Depending on the sensor's current value it's state could be:
  - Lower Non-Recoverable
  - Lower Critical
  - Lower Non-Critical
  - Upper Non-Critical
  - Upper Critical
  - Upped Non-Recoverable

The ***local_sensor*** command can also be used to display a a more detailed description for a single sensor:

- name
- type
- value
- threshold values
- sensor state
- outputs assigned to threshold events
- fan control groups the sensor is assigned to (only for temperature sensors)
- active level for discrete sensors
- output function for output sensors
- active drivers for output sensors

For the complete syntax of the commands refer to the chapter **5. Command Line interface(CLI).**

2.2 Local Sensor Numbers

This section refers to the local sensors implemented on the Shelf Manager. The correspondence between the sensor numbers and monitored parameters is defined in a table presented bellow.

Depending on the monitored system the ShMC can be set up differently. Not always all the sensors described bellow are available. To check out all the installed sensors you can use the ***local_sensor*** command.

The ShMC monitors only the parameters for which a SDR has been uploaded. The monitored parameters set can be changed by uploading a new SDR set. The SDRs are encapsulated in a configuration file. Besides the SDRs, the config file also hosts several other ShMC parameters.

Sensor Number	Monitored System Parameter
1	reserved
2	V0 Voltage
3	V1 Voltage
4	V2 Voltage
5	V3 Voltage
6	V4 Voltage
7	V5 Voltage
8	V6 Voltage
9	V7 Voltage
26	Temperature sensor 1
27	Temperature sensor 2
28	Temperature sensor 3
29	Temperature sensor 4
30	Temperature sensor 5
31	Temperature sensor 6
37	Tachometer signal for Fan1
38	Tachometer signal for Fan2
39	Tachometer signal for Fan3
40	Tachometer signal for Fan4
41	Tachometer signal for Fan5

42	Tachometer signal for Fan6
43	Tachometer signal for Fan7
44	Tachometer signal for Fan8
45	Tachometer signal for Fan9
46	Tachometer signal for Fan10
47	Tachometer signal for Fan11
48	Tachometer signal for Fan12
64	Digital Input 1
65	Digital Input 2
66	Digital Input 3
67	Digital Input 4
68	Digital Input 5
69	Digital Input 6
70	Digital Input 7
71	Digital Input 8
72	Digital Input 9
73	Digital Input 10
74	Digital Input 11
75	Digital Input 12
76	Digital Input 13
77	Digital Input 14
78	Digital Input 15
79	Digital Input 16
80	Digital Output 1
81	Digital Output 2
82	Digital Output 3
83	Digital Output 4
84	Digital Output 5
85	Digital Output 6
86	Digital Output 7
87	Digital Output 8
88	Digital Output 9
89	Digital Output 10
90	Digital Output 11
91	Digital Output 12
92	Digital Output 13
93	Digital Output 14
94	Digital Output 15
95	Digital Output 16
97	ShMC Power On

Table 2: Local Sensor Numbers

2.3 Sensor Event Log (SEL)

The ShMC uses a non volatile Flash memory to log the events generated by the sensor monitoring function. The Log can hold information for up to 65534 events.

! After the log reaches full capacity, a warning message will be displayed and all new events will be disregarded ! New Events will be logged only after the sel is cleared.

Logging is enabled for each event individually using the SDRs. The ShMC uses the ***Assertion Events Mask*** and ***Deassertion Events Mask*** fields in each SDR to decide which events are logged.

! Only events that are supported by the sensor, and are accordingly marked by the SDR will be logged.

For each event the log provides the following information:

- time stamp
- number and name of the sensor that generated the event
- event type: - for threshold sensors: UNR,UC,UNC,LNC,LC,LNR
  - for discrete sensors: 1 (Asserted), 0 (De-Asserted)
- sensor value that triggered the event and threshold value (only for threshold sensors)

Example 1. Sensor event log

```
 %>sel print
------------------------------------------------------------------------------
                             Sensor  Event Log

Rec.ID. dd.mm.yyyy hh:mm:ss  Sensor No. Name             Event  Ev.Dir  Value  Threshold
------------------------------------------------------------------------------
------------------------------------------------------------------------------
0x0001  01.01.2012 00:00:00     97       SHMC Power ON     1 (Asserted)
0x0002  01.01.2012 00:00:00     2        V0                LNC     As      0.00     2.86
0x0003  01.01.2012 00:00:00     4        V2                LC      As      0.00     11.38
0x0004  01.01.2012 00:00:00     6        V4                LC      As      0.00     2.16
0x0005  01.01.2012 00:00:00     8        V6                LC      As      0.00     0.24
0x0006  01.01.2012 00:00:00     3        V1                LC      As      0.00     4.76
0x0007  01.01.2012 00:00:00     5        V3                UC      As      2.58     -11.34
0x0008  01.01.2012 00:00:00     7        V5                LC      As      0.00     0.66
0x0009  01.01.2012 00:00:00     9        V7                LC      As      0.00     12.39
```

Each ShMC restart is marked by an event 1 (Asserted) for the *"ShMC Power On Sensor".*
For more details and syntax refer to the ***sel*** command section of this user manual.

3 IPMB interface

The Shelf Manager was developed based on the IPMI v1.5 specification.

3.1 Supported command list

IPM Device "Global" Commands	NetFn	CMD
Get Device ID	App	01h
Cold Reset	App	02h
Get Self Test Results	App	04h
Get ACPI Power State	App	06h
Send Message	App	34h
Get Channel Authentication Capabilities	App	38h
Get Session Challenge	App	39h
Activate Session	App	3Ah
Set Session Privilege Level	App	3Bh
Close Session	App	3Ch
Get Session Info	App	3Dh
Chassis Commands	**NetFn**	**CMD**
Get Chassis Capabilities	Chassis	00h
Get Chassis Status	Chassis	01h
Chassis Control	Chassis	02h
Event Commands	**NetFn**	**CMD**
Set Event Receiver	S/E	00h
Get Event Receiver	S/E	01h
Platform Event	S/E	02h
Sensor Device Commands	**NetFn**	**CMD**
Get Device SDR Info	S/E	20h
Get Device SDR	S/E	21h
Reserve Device SDR Repository	S/E	22h
Set Sensor Hysteresis	S/E	24h
Get Sensor Hysteresis	S/E	25h
Set Sensor Threshold	S/E	26h
Get Sensor Threshold	S/E	27h
Set Sensor Event Enable	S/E	28h
Get Sensor Event Enable	S/E	29h
Get Sensor Reading	S/E	2Dh
FRU Device Commands	**NetFn**	**CMD**
Get FRU Inventory Area Info	Storage	10h
Read FRU Data	Storage	11h
Write FRU Data	Storage	12h

SDR Device Commands	NetFn	CMD
Get SDR Repository Info	Storage	20h
Reserve SDR Repository	Storage	22h
Get SDR	Storage	23h
SEL Device Commands	**NetFn**	**CMD**
Get SEL Info	Storage	40h
Reserve SEL	Storage	42h
Get SEL Entry	Storage	43h
Add SEL Entry	Storage	44h
Delete SEL Entry	Storage	46h
Clear SEL	Storage	47h
Get SEL Time	Storage	48h
Set SEL Time	Storage	49h
PICMG Commands	**NetFn**	**CMD**
Get PICMG Properties	PICMG	00h
Get Address Info	PICMG	01h
Get Shelf Address Info	PICMG	02h
Set Shelf Address Info	PICMG	03h
FRU Control	PICMG	04h
Get FRU LED Properties	PICMG	05h
Get LED Color Capabilities	PICMG	06h
Set FRU LED State	PICMG	07h
Get FRU LED State	PICMG	08h
Set IPMB State	PICMG	09h
Set FRU Activation Policy	PICMG	0Ah
Get FRU Activation Policy	PICMG	0Bh
Set FRU Activation	PICMG	0Ch
Get Device Locator Record ID	PICMG	0Dh
Get Fan Speed Properties	PICMG	14h
Set Fan Level	PICMG	15h
Get Fan Level	PICMG	16h

3.2 Shutting down a CPCI system

The ShMC can be used to safely turn off a CPCI system. The Shutdown and Power On processes are controlled either by a front panel push button or by the one available on the web page.

When a shutdown is commanded, the ShMC informs the boards of the current request. After all boards deactivate their OS and are ready to be turned off, the ShMC disables the power supply for the chassis, using the configured shutdown output pin. The chassis can also be turned back on, using the designated shutdown push button or the web page control. As CPCI boards do not require activation, the power on process resumes at using the configured shutdown output to enable the chassis power supply.

The soft shutdown of the boards consists of ACPI (*Advanced Configuration and Power Interface Specification*) related IPMI commands:

- *Get ACPI State*
- *Chassis Control*

  For the Chassis Control command only the Soft Shutdown and Power Down options are used.

  The board shutdown IPMI communication protocol is depicted in the following figure.

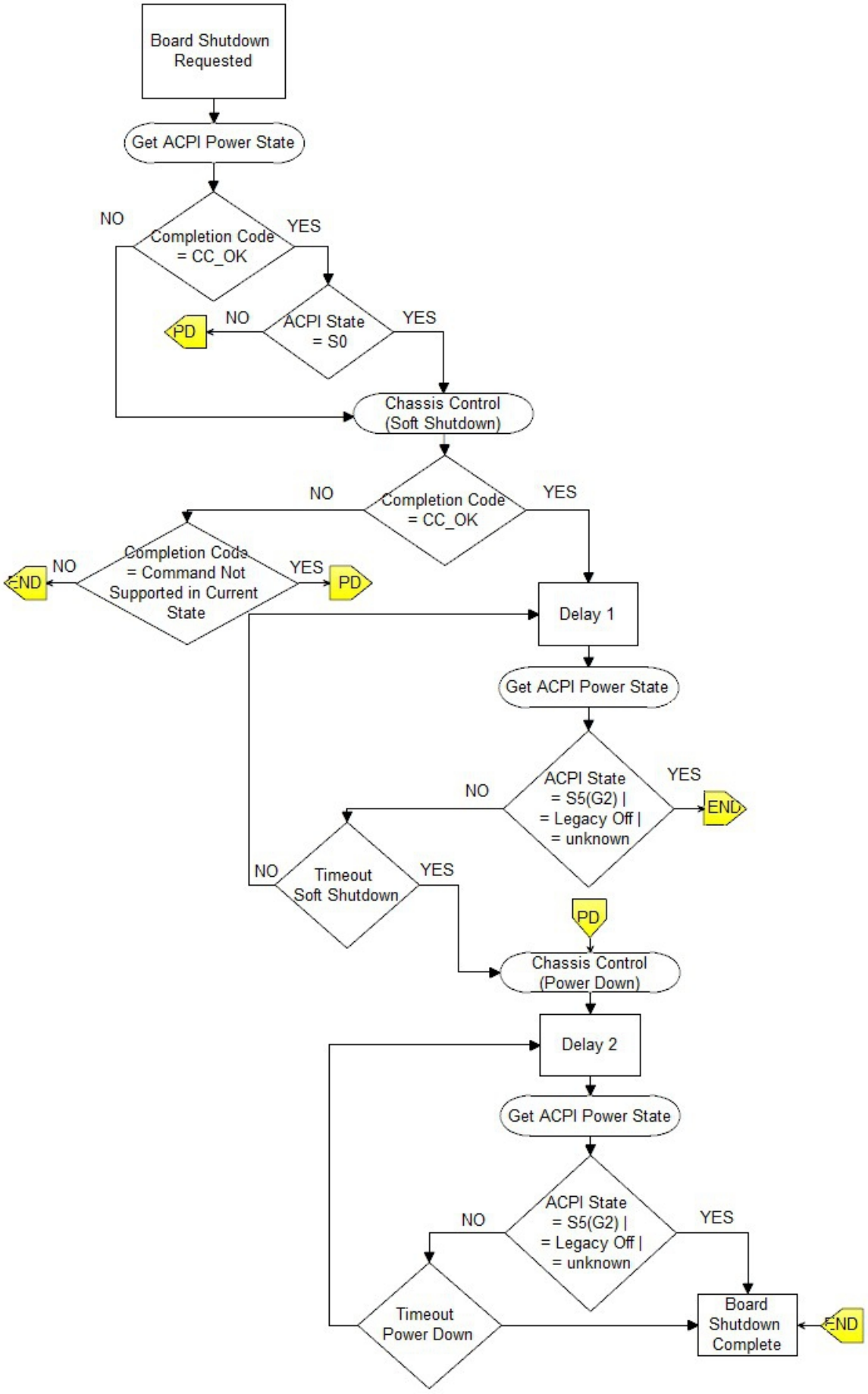

Figure 2: Board Shutdown IPMI Protocol

4 Ethernet interface

The integrated 10/100Mbps Ethernet interface allows the ShMC to be linked to any existing network. The interface supports DHCP, SNMP, TFTP, HTTP and TELNET protocols via TCP/IP and UDP.

All monitored system parameters can be displayed via a standard browser (HTTP protocol).

The Command Line Interface (CLI) is accessible via TELNET, allowing remote control of the ShMC.

The use of standard protocols avoids the need for special software or drivers and so achieves platform-independence. The TCP/IP protocol supports up to 10 simultaneous connections.

! The factory default setting for the ShMC is DHCP enabled so it negotiates automatically all the necessary addresses. If a fixed IP address is desired, DHCP must be disabled and the address has to be set manually. For all these operations the ***lanconfig*** command needs to be used.

Terminal settings:

- Local echo: *off*
- Local line editing: *off*
- Backspace key: *Control-H*

4.1 WEB

The Shelf Manager includes a built in WEB server that provides a simple way to access the management information.

4.1.1 Overview

The WEB page can be designed as a graphical representation of the monitored System Platform, thus providing a very intuitive way of obtaining system / board information.(Figure 2 exemplifies the web page for a 5 slot ATCA system) . The available information includes:

- Board / Carrier /Shelf Field Replaceable Unit (FRU) information file (Manufacturer's Name, Part number, Serial Number, Board Connectivity Records)
- Sensors information: value, name,measuring unit, status,threshold and hysteresis values
- System Event Log (SEL) : sensor events
- ShMC attributes: MAC address, Serial number, Firmware version

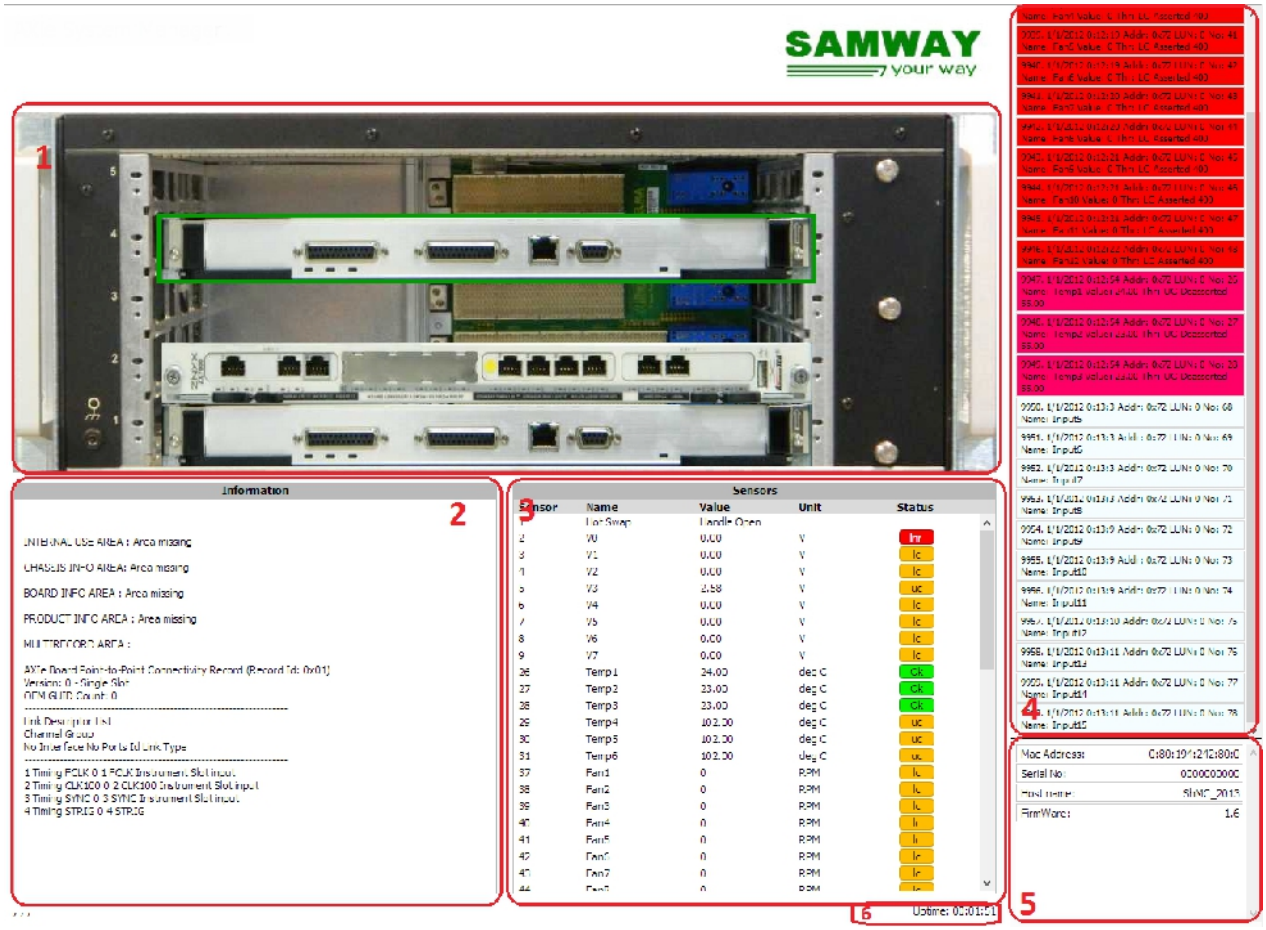

Figure 3: Web Interface overview

The default Layout for the Web Page is composed of several areas:

1. *Dynamic System Platform Representation*:: The images are populated or removed depending on the hot-swap state of the boards. Both system platform and boards are treated as objects and can be selected using a mouse click. The information in panes *2. FRU info* and *3. Sensor info* is changed depending on the selected object.
2. *FRU info*: displays the FRU information for the selected object(board or system platform)
3. *Sensor info:* displays the values, names and status for all the sensors of the selected object. All the sensors are considered objects and can be selected using a mouse click.
4. *System Event Log (SEL)*: displays all the sensor events received by the Shelf manager starting from the moment the Web page has been loaded.
5. *Info Area*: displays more details for the selected sensor (threshold and hysteresis values) or, if no sensor is selected, displays ShMC attributes: MAC address, Serial number, Firmware version.
6. *Uptime :* displays the amount of time the ShMC has been operational. It is reset at each ShMC restart.

4.1.2 Architecture

The Web Server uses a Representational State Transfer (REST) based architecture and Extensible Markup Language (XML) files.

REST is an architectural style that abstracts the architectural elements within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements. REST has emerged as a predominant web API design model.

XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The design goals of XML emphasize simplicity,generality, and usability over the Internet.

The ShMC WEB page is composed of:

- objects : board and system images; sensors
- information display areas

When an object is selected by a mouse click, the web browser requests predefined XML files using predefined commands. The information contained by the XML files returned by the ShMC is parsed in the information display windows.

The XML files could also be requested by an application software in case a custom managing solution is required. The format of the XML files is described in the following chapter.

4.1.3 REST Commands

"/settings"

For this request the ShMC returns an XML file containing the following tags:

- <mac_addr>
- <serial_no>
- <host_name>
- <firmware> : firmware version
- <uptime> : the amount of time the ShMC has been operational
  - <H>: hours
  - <M>: minutes
  - <S>: seconds

Request: GET **/settings**
IP Address/**settings** (ex: 192.168.16.1/settings)

Response:

```
<?xml version="1.0"?>
<settings>
<mac_addr>0:80:194:242:80:0</mac_addr>
<serial_no>0000000000</serial_no>
```

```
<host_name>ShMC_2013</host_name>
<firmware>1.6</firmware>
<uptime>
        <H>0</H>
        <M>31</M>
        <S>21</S>
</uptime>
</settings>
```

"/frustatus"

For this request the ShMC returns an XML file that contains the following tags:

- <boot_cnt>: boot index. The index is incremented at each ShMC start-up.
- <sel_cnt>: System Event Log (SEL) index. The index is incremented when an event is added to the SEL. This field is used to detect if new events have occurred.
- <fru_list>: a list with all the modules(FRUs) present in the system(including the ShMC)
- <fru_addr>: the IPMB address of the module
- <fru_id>: active FRU ID. The Module Management Controller (MMC) is always represented by FRU ID 0 and is always present for active modules. Additional FRU IDs can also be active.

Request: GET /**frustatus**
IP Address/**frustatus** (ex: 192.168.16.1/frustatus)
Response:

```
<?xml version="1.0"?>
<fru_status>
<boot_cnt>315</boot_cnt>
<sel_cnt>9791</sel_cnt>
<fru_list>
        <fru_addr addr="0x00">
                <fru_id>0</fru_id>
        <fru_addr addr="0x82">
                <fru_id>0</fru_id>
        </fru_addr>
        <fru_addr addr="0x84">
                <fru_id>0</fru_id>
        </fru_addr>
        <fru_addr addr="0x86">
                <fru_id>0</fru_id>
        </fru_addr>
        <fru_addr addr="0x10">
                <fru_id>0</fru_id>
        </fru_addr>
```

```
</fru_list>
</fru_status>
```

"/sel/start_index/end_index"

This command retrieves multiple SEL event records. The command use two indexes (start,end) to define the desired number of records. The ShMC returns an XML file that contains all the event records that have an index between the start index and the end index. The XML file uses the following tags:

- <rec id> : the index of the current record
- <tmp> : the time when the event was triggered
- <addr>: address of the card that launched the event
- <lun>: the LUN on which the sensor resides
- <no>: sensor number
- <name>: sensor name
- <type>: code representing the sensor type
- <sta>: sensor state, available only for discrete sensors
- <ev_type>: threshold that triggered the event for threshold sensors: UNR (upper non-recoverable), UC(upper critical), UNC(upper non-critical), LNC (lower non-critical), LC(lower critical), LNR(lower non-recoverable)
- <ev_dir> Asserted, DeAsserted
- <val>: sensor value
- <thr>: threshold value

Request: GET **/sel/***start_index/end_index*
IP Address**/sel/***start_index/end_index* (ex: 192.168.16.1/sel/1/2)
Response:

```
<?xml version="1.0"?>
<sel>
<rec id="1">
        <tmp>1325376000</tmp>
        <addr>0x72</addr>
        <lun>0</lun>
        <no>97</no>
        <name>SHMC Power ON</name>
        <type>192</type>
        <sta>0</sta>
</rec>
<rec id="2">
        <tmp>1325376000</tmp>
        <addr>0x72</addr>
        <lun>0</lun>
        <no>2</no>
        <name>V0</name>
        <type>2</type>
        <ev_type>LNC</ev_type>
```

```
            <ev_dir>Asserted</ev_dir>
            <val>0.00</val>
            <thr>2.86</thr>
        </rec>
        </sel>
```

"/fruinfo/*FRU_Address*/*FRU_Id*"

This command retrieves a **text file** containing the FRU information for the desired FRU Id on the specified card. The command uses two parameters:

- *FRU_Address*: the IPMB address of the card(hexadecimal value)
- *FRU_Id*: used to distinguish between multiple FRUs located on the same card. The FRU Id for the Module Management Controller (MMC),(the card itself) is 00.

Request: GET **/fruinfo**/*FRU_Address/FRU_Id*
IP Address/**fruinfo**/*FRU_Address/FRU_Id*(ex: 192.168.16.1/fruinfo/0x82/0)
Response: *text file*

"/sensor/FRU_Address/FRU_Id"

This command retrieves all the sensor associated to a FRU Id of a card. The command uses two parameters:

- *FRU_Address*: the IPMB address of the card(hexadecimal value)
- *FRU_Id*: used to distinguish between multiple FRUs located on the same card. The FRU Id for the Module Management Controller (MMC),(the card itself) is 00.

If this parameter is missing the ShMC will return the sensors associated to FRU Id 0
The ShMC responds to this request using an XML file that contains the following tags:

<name>:sensor name
<value>:sensor value
<unit>:sensors unit of measurement
<state>:sensor state (lnr, lc, lnc, unc, uc, unr)

Request: GET **/sensor**/*FRU_Address*/*FRU_Id*
IP Address***/sensor***/FRU_Address/FRU_Id*(ex: 192.168.16.1/sensor/0x82)
Response:

```
<?xml version="1.0"?>
<sensor_list>
    <sensor no="1">
        <name>Hot Swap</name>
        <value>Handle Open </value>
    </sensor>
    <sensor no="2">
        <name>V0</name>
        <value>0.00</value>
        <unit>V</unit>
        <state>lnr</state>
    </sensor>
</sensor_list>
```

"/sdr/FRU_Address/sensor_No"

This command retrieves additional information for a specific sensor on the desired card. The command uses two parameters:

- *FRU_Address*: the IPMB address of the card(hexadecimal value)
- *sensor_No*: sensor number for the desired sensor

The ShMC responds to this request using an XML file that contains the following tags:

- <name>
- <entity_id> : entity id for the sensor's owner
- <entity_instance> :entity instance for the sensor's owner
- <unr>: upper non-recoverable threshold value, if the unr threshold is enabled
- <uc>: upper critical threshold value, if the uc threshold is enabled
- <unc>: upper non-critical threshold value, if the unc threshold is enabled

- <lnc>: lower non-critical threshold value, if the lnc threshold is enabled
- <lc>: lower critical threshold value, if the lc threshold is enabled
- *<lnr>: lower non-recoverable threshold value, if the lnr threshold is enabled*
- <hyst_pos>: positive going hysteresis value
- <hyst_neg>: negative going hysteresis value
- <nominal_reading>
- <normal_maximum>
- <normal_minimum>
- <maximum_reading>
- <minimum_reading>

Request: GET **/sdr**/*FRU_Address/sensor_No*
IP Address ***/sdr****/FRU_Address/sensor_No(ex: 192.168.16.1/sensor/0x10/5)*
Response:

```
<?xml version="1.0"?>
<sensor no="5">
        <name>V3</name>
        <entity_id>0x01</entity_id>
        <entity_instance>0x61</entity_instance>
        <uc>-11.34</uc>
        <lc>-12.64</lc>
        <hyst_pos>0.07</hyst_pos>
        <hyst_neg>0.07</hyst_neg>
        <nominal_reading>-14.00</nominal_reading>
        <normal_maximum>-14.00</normal_maximum>
        <normal_minimum>-14.00</normal_minimum>
        <maximum_reading>2.58</maximum_reading>
        <minimum_reading> -14.00</minimum_reading>
</sensor>
```

"/picture/FRU_Address"

This command retrieves the pictures for the boards and system platform.

- *FRU_Address*: the IPMB address of the card(hexadecimal value). For the picture of the System platform the request uses FRU_Address = 0.

Request: GET /**picture**/*FRU_Address*
IP Address ***/picture****/FRU_Address*(192.168.16.1/picture/0x80; 192.168.16.1/picture/0)
Response: picture file

4.1.4 How it all works

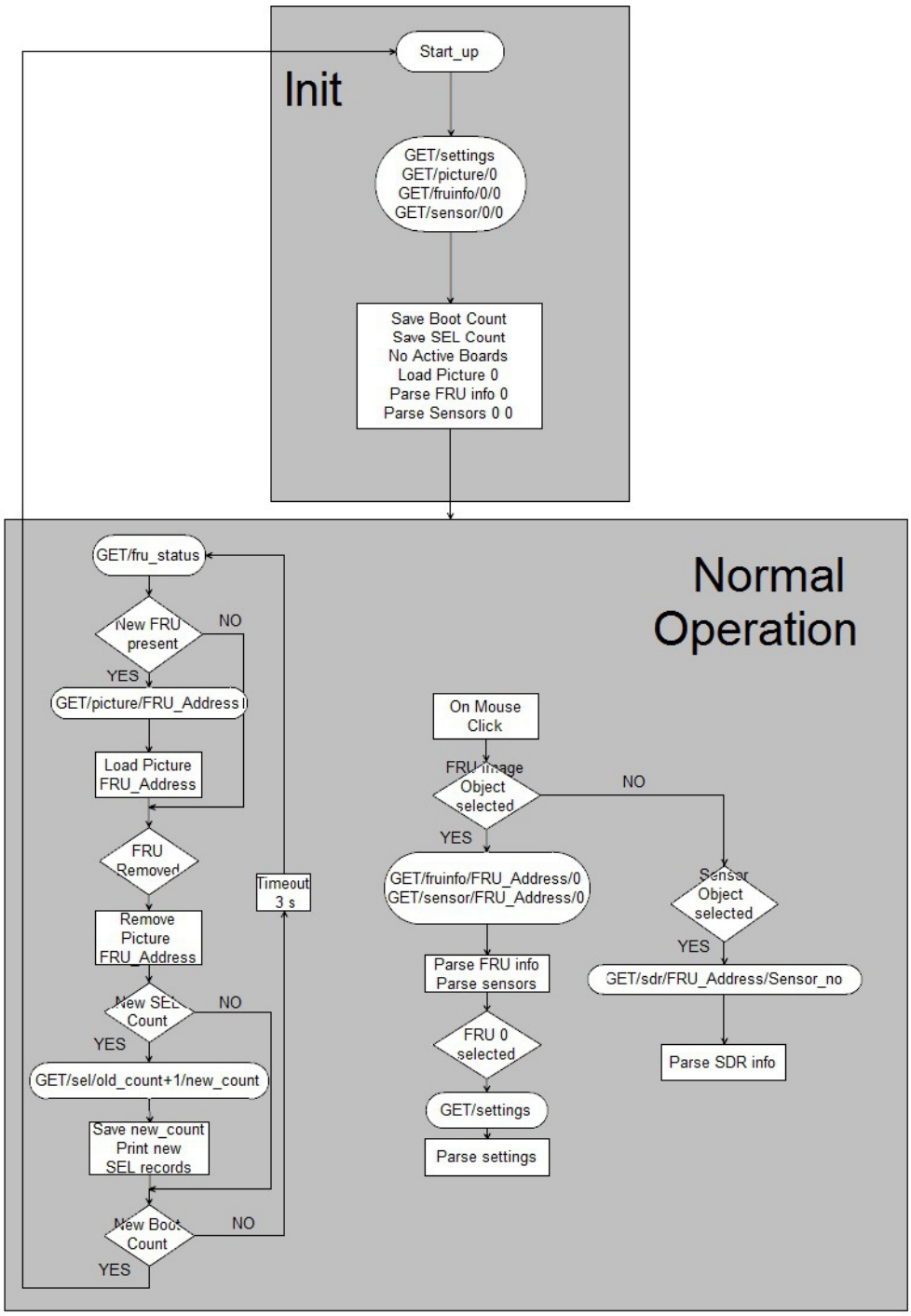

Figure 4: Default Web Page Functional Diagram

Figure 3 describes the functional diagram of the default Web Page. The web page has 2 operational states: Initialization and Normal Operation.

At start-up, and each time it is re-initialized, the web page is empty. The Web page requests and saves the ShMC settings (**GET/settings**): MAC Address, Serial Number, Firmware Version,Uptime. The page also saves some parameters from the response of the frustatus request: Boot count, SEL count.

Next, the web page request information for the system platform(FRU 0): picture, FRU info,

sensors. (**GET/picture/0, GET/fruinfo/0/0, GET/sensor/0/0**). Once the information is received, the web page displays it.

After the system platform is initialized the init phase ends. By default all boards are considered absent during this phase.

The normal operation phase of the web page is split in two processes:

- A periodical process that check to see if new boards have been inserted, or if boards have been removed
- A mouse click listener that changes the active object

The periodic process issues a **GET/frustatus** request at every 3 seconds. By comparing the FRU list in the response and the FRUs loaded, the page determines if boards have been inserted or removed. For all the new boards the web page sends a **GET/picture/FRU_Address** request and loads the received picture. For all the boards that have been removed, the web page also removes the picture.

Using the response for the **frustatus** request the web page also checks if new SEL events have occurred. If the SEL count from the response is different from the one saved by the page, a **GET/sel/old_count+1/new_count** request is sent. Using the response the web page displays all the new SEL events.

The **frustatus** request is also used to detect restarts of the ShMC. If the Boot count in the response is different than the saved one, the WEB page is restarted: all pictures are removed and all the internal variables are reinitialized.

The WEB page uses a mouse click listener to determine if the current selected object has been changed. The WEB page uses two types of objects:

- FRU images
- FRU sensors

At each mouse click the WEB page determines if the active object has been changed.

If the active FRU image object has been changed, the WEB page request the FRU info and sensors for the new active object: **GET/fruinfo/FRU_Address/0**, **GET/sensor/FRU_Address/0**. Using the responses of the requests, the web page updates the information visible in the *FRU Info* and *Sensor Info* panels. By default, at start-up, the active image object is the system platform (FRU 0).

If the active sensor object has been changed, the WEB page request the detailed information or the new active sensor: **GET/sdr/FRU_Address/Sensor_No.** Using the response for the sdr request, the WEB page updates the information in the *Info Area* panel. By default, at start-up no sensor object is selected and the Info are panel displays ShMC parameters: MAC address, Serial No., Firmware version.

After the first sensor object becomes active, the *Info Area* panel will display sensor information. To return to displaying ShMC parameters, a click on the system platform image is necessary.

4.2 Telnet

The Telnet interface supports two operating modes: - Command Line Interface (CLI) and Serial Over Lan (SOL).

The CLI is accessible if the operator is logged on as "user" or "admin" profile.

If the operator is logging into the ShMC as "serial" profile the SOL mode is activated. Whenever the ShMC enters in SOL mode the behavior of RS232 port will change. All the data received on the serial port will be transmitted in telnet packets over Ethernet to all telnet connections opened on "serial" profile. The data received from the telnet connections opened in "serial" mode is transferred to the serial port. This feature allows remote control of a CPU card inside the system. For this the CPU terminal screen should be redirected to the serial port and the serial

port of the CPU needs to be connected with the serial port of the ShMC.

Default access settings:

login: serial
password: SERIAL

In SOL mode the SysRq key could be send by sending a Telnet break.

Note: If SOL mode is used it is recommended to make the SOL mode as default using "sol" command.

4.3 SNMP

The ShMC supports SMNP v1, v2c and SNMP v3 with authentication and encryption. The software supports MD5 for authentication, while for privacy supports DES and AES.
For application where data security is an issue, SNMP v3 with authentication and encryption could be used. For such applications the telnet and the web interfaces could be disabled to prevent not secured connections to the ShMC. To disable the Telnet or Web interface use the "eth" command or write the SNMP variables telnet and web to 0 in the control node.
When the SNMP is configured to use authentication in version 3 (see "snmp" command) and the authentication password is not empty the SNMP v1 and V2C are disabled. Only SNMP v3 authenticated messages are accepted.

The default SNMP v3 passwords are:
"SnmpUserAuth" – for SNMP user authentication
"SnmpUserPrivacy" – for SNMP user privacy
"SnmpAdminAuth" – for SNMP admin authentication
"SnmpAdminPrivacy" – for SNMP admin privacy

The authentication and encryption is disabled by default on both user and admin profiles so that the ShMC accepts SNMP v1, SNMP v2c and unauthenticated SNMP v3 messages. Te enable authentication use "snmp" command.

The SNMP data is structured in some categories:

.system

- contains information about the ShMC:name, part number, serial number, MAC address and software version

.temp

- contains the number of temperature sensors and for each sensor, the name, value, state, thresholds and hysteresis.

.voltage

- contains the number of voltage sensors and for each sensor, the name, value, state, thresholds and hysteresis

.current

- contains the number of current sensors and for each sensor, the name, value, state, thresholds and hysteresis

.fan

- contains the number of fan sensors, the speed level of each group of fans and for each sensor, the name, value, state, thresholds and hysteresis

.input

- contains the number of inputs and for each input, the name and state

.output

- contains the number of outputs and for each output, the name and state

.control

- this category allows different configurations to be done to the ShMC. The RS232 speed and the user and admin passwords could be read or changed, the Telnet and Web interfaces could be enabled /disabled. The configuration settings could be saved and the ShMC could be restarted.

4.4 RMCP

The ShMC supports Remote Management Control Protocol (RMCP). The RMCP connections requires authentication. The supported authentication protocol is MD5. Two user names are accepted for RMCP connections: "user" – with "User" privilege level and "admin" with "Administrator" privilege level. The user names and privilege levels are fixed, they cannot be changed through IPMI commands. The password used are the same passwords configured for Telnet, CLI, SNMP v1 and v2c access. The default passwords are: "USER" – for "user" profile and "ADMIN" - for admin profile.

5 RS232 serial interface

The ShMC provides an RS232 serial interface through which the commands of the Command Line Interface (CLI) can be sent.

On Windows systems, we recommend the use of "TeraTerm" or "Hyperterminal" as the terminal programs.

Default Terminal settings:

- 19200 bits per second
- data bits: 8
- parity: none
- stop bit: 1

The baud rate of the RS232 Serial Interface can be changed. The available baud rates are 9600, 19200, 38400,115200. To change the speed of the Serial Interface use the ***scispeed*** command

In addition, the ***xmodem*** command can be used via the serial interface for file transfer.

Use a 1:1 serial cable for direct connection to the serial port of a PC.
When using ***xmodem*** in "Hyperterminal" the transfer of the desired file can take up to 10 seconds to start.

6 Command Line Interface (CLI)

The Command Line Interface (short-form: CLI) is available via both Telnet (**3 Ethernet interface**) and the RS232 serial interface (**4 RS232 serial interface**).

The user can read or newly configure and save system parameters via the CLI.

Access is divided into 2 profiles and is password-protected.

"user" profile:
System parameters can only be read in this profile – the exception to this write-protect is the ***lanconfig*** command for setting the IP, subnet and gateway addresses.

"admin" profile:
Full access to all system parameters is granted. All available CLI commands can be executed. To avoid possible damage or malfunctions, the access data for this profile must only be available to trained personnel with appropriate knowledge and competence relating to the system in which the ShMC is used!
The profiles can be changed using the ***logout*** command.

The measured values are available at any time via the RS232 serial interface and via Telnet. In addition, limits and system parameters can be changed at any time with the unit in service. As soon as you have established a connection, you will be prompted to log in.

Default access settings:

login: user
password: USER

login: admin
password: ADMIN

! The passwords can be changed using the ***passw*** command. The passwords can also be disabled by changing them to an empty string.

General syntax conventions

Command [parameter1 | parameter2 | parameter3 [*value*]]

[ ] = optional
A command on its own with no entry of other parameters returns all available current values associated with the command.
The parameters separated by "|" rule out each other.
If the command line contains a *value*, this is assigned to the corresponding parameter and saved temporarily in the RAM. The change is active immediately. If the new value is desired to be valid after the reboot, the environment variables must be saved with the ***saveenv*** command. Changes not confirmed with ***saveenv*** are lost after a reboot.

6.1 voltage command

Syntax: voltage

Functions:
Displays information about all the installed voltage sensors. For each installed sensor the command displays:

- sensor number
- sensor name
- sensor type
- current value
- measuring unit
- current state

Example 1. voltage sensor values read out

```
%>voltage

----------------------------Sensor List-----------------------------
--no--Name-------------Type----Value--Unit---State-----------------

*   2  V0               Thr     1.00   V      Lower Non-Recoverable
*   3  V1               Thr     2.30   V      Ok
*   4  V2               Thr     4.55   V      Lower Critical
```

6.2 fan command

Syntax: fan

Functions:

Displays information about all the installed fan speed sensors. For each installed sensor the command displays:

- sensor number
- sensor name
- sensor type
- current value
- measuring unit
- current state

Example 1. fan sensor values read out

```
%>fan
----------------------------Sensor List---------------------------
--no--Name-------------Type--Value--Unit---State-----------------
* 37  Fan1             Thr   8800   RPM    Ok
* 38  Fan2             Thr   2700   RPM    Ok
* 39  Fan3             Thr   2700   RPM    Ok
* 40  Fan4             Thr   2600   RPM    Ok
* 41  Fan5             Thr   4800   RPM    Ok
* 42  Fan6             Thr   8800   RPM    Ok
```

6.3 temp command

Syntax: temp

Functions:

Displays information about all the installed temperature sensors. For each installed sensor the command displays:

- sensor number
- sensor name
- sensor type
- current value
- measuring unit
- current state

Example 1. temperature sensor values read out

```
%>temp
----------------------------Sensor List-----------------------------------
--no--Name-------------Type--Value--Unit---State---------------------------
* 26  Temp1            Thr   26.00  deg C  Ok
* 27  Temp2            Thr   26.00  deg C  Ok
* 28  Temp3            Thr   26.00  deg C  Ok
* 29  Temp4            Thr   Not Present!
* 30  Temp5            Thr   Not Present!
* 31  Temp6            Thr   Not Present!
```

6.4 input command

Syntax: input

Functions:
Displays information about all the input sensors. For each sensor the command displays:

- sensor number
- sensor name
- sensor type
- current value

Example 1. Input sensors values read out

```
%>input
-----------------------------Sensor List-----------------------------
--no--Name--------------Type----Value--Unit---State------------------

*  64  Input1            Input    1   (Asserted)
*  65  Input2            Input    0   (De-Asserted)
*  66  Input3            Input    0   (De-Asserted)
*  67  Input4            Input    0   (De-Asserted)
*  68  Input5            Input    0   (De-Asserted)
*  69  Input6            Input    1   (Asserted)
*  70  Input7            Input    1   (Asserted)
*  71  Input8            Input    1   (Asserted)
*  72  Input9            Input    1   (Asserted)
*  73  Input10           Input    1   (Asserted)
*  74  Input11           Input    0   (De-Asserted)
*  75  Input12           Input    0   (De-Asserted)
*  76  Input13           Input    0   (De-Asserted)
*  77  Input14           Input    1   (Asserted)
*  78  Input15           Input    1   (Asserted)
*  79  Input16           Input    1   (Asserted)
```

6.5 output command

Syntax: output

Functions:
Displays information about all the output sensors. For each sensor the command displays:

- sensor number
- sensor name
- sensor type
- current value

Example 1. Output sensors values read out

```
%>output
-----------------------------Sensor List-----------------------------
--no--Name--------------Type----Value--Unit---State------------------
*  80  Output1           Output   1   (Asserted)
*  81  Output2           Output   1   (Asserted)
*  82  Output3           Output   0   (De-Asserted)
*  83  Output4           Output   0   (De-Asserted)
*  84  Output5           Output   0   (De-Asserted)
*  85  Output6           Output   1   (Asserted)
*  86  Output7           Output   0   (De-Asserted)
*  87  Output8           Output   1   (Asserted)
*  88  Output9           Output   0   (De-Asserted)
*  89  Output10          Output   0   (De-Asserted)
*  90  Output11          Output   0   (De-Asserted)
*  91  Output12          Output   1   (Asserted)
*  92  Output13          Output   0   (De-Asserted)
*  93  Output14          Output   0   (De-Asserted)
*  94  Output15          Output   0   (De-Asserted)
*  95  Output16          Output   0   (De-Asserted)
```

6.6 local_sensor command

Syntax: local_sensor [*sensor_no* **[threshold** threshold_code *value* | *disable*] **|**
[hysteresis hysteresis_code *value*] **|**
[output [threshold_code] hex_outputs] **|**
[activelevel 0|1] **|**
[fancontrol fancontrol_mask] **|**
[output_type OR | AND] **|**
[input_type mswitch | on_off]] **|**
[assert | deassert] **|**
[userpermission allow|deny]
[timeout *time_value*]
[debounce *debounce_value*]

- sensor_no : the local sensor number that identifies a particular sensor located on the Shelf Manager (see **2.2 . Sensor Numbers**)
- threshold_code is defined in Table 3: Threshold_Code
- hysteresis code is defined in Table 4: Hysteresis_Code
- hex_outputs defined in Table 5: Hex_Outputs
- fancontrol_mask defined in Table 6: Fancontrol_Mask
- time_value :0..15
- debounce_value :0..255

Threshold	threshold_code
Lower Non-Recoverable	lnr
Lower Critical	lc
Lower Non-Critical	lnc
Upper Non-Critical	unc
Upper Critical	uc
Upper Non-Recoverable	unr

Table 3: Threshold_Code

Hysteresis	*hysteresis_code*
Negative going Hysteresis Value	neg
Positive going Hysteresis Value	pos

Table 4: Hysteresis_Code

hex_outputs bit	Value
15	Output 16
14	Output 15
13	Output 14
12	Output 13
11	Output 12

10	Output 11
9	Output 10
8	Output 9
7	Output 8
6	Output 7
5	Output 6
4	Output 5
3	Output 4
2	Output 3
1	Output 2
0	Output 1

Table 5: Hex_Outputs Bit Mask

fancontrol_mask bit	Bit value	Details
7 .. 3	0	Reserved
2	Sensor active state for PWM 3	1: sensor active 0: sensor inactive
1	Sensor active state for PWM 2	1: sensor active 0: sensor inactive
0	Sensor active state for PWM 1	1: sensor active 0: sensor inactive

Table 6: Fancontrol_Mask

Functions:

The local_sensor command can be used either to access information about the Shelf Manager local sensors, or to change different sensor parameters.

If the command is used without any parameter, it will return basic information about all the active sensors. For each installed sensor the command displays:

- local sensor number
- sensor name
- sensor type
- current value
- measuring unit (only for threshold sensors)
- current state (only for threshold sensors)

Example 1. Read all the active sensors

```
%>local_sensor
----------------------------Sensor List---------------------------
--no--Name------------Type--Value--Unit---State-----------------
*  2  +3.3V           Thr   3.27   V      Ok
*  3  +5V             Thr   5.04   V      Ok
*  4  +12V            Thr   0.00   V      Lower Critical
*  5  -12V            Thr   -11.88 V      Ok
* 26  Temp1           Thr   27.00  deg C  Ok
* 27  Temp2           Thr   27.00  deg C  Ok
* 45  Fan9            Thr   2100   RPM    Ok
* 46  Fan10           Thr   2100   RPM    Ok
* 47  Fan11           Thr   2100   RPM    Ok
* 48  Fan12           Thr   2100   RPM    Ok
* 64  Input1          Input   1  (Asserted)
* 65  Input2          Input   0  (De-Asserted)
* 66  Input3          Input   0  (De-Asserted)
* 67  Input4          Input   0  (De-Asserted)
* 68  Input5          Input   0  (De-Asserted)
```

```
* 80  Output1          Output  1   (Asserted)
* 81  Output2          Output  1   (Asserted)
* 82  Output3          Output  0   (De-Asserted)
* 83  Output4          Output  0   (De-Asserted)
```

If only the sensor number parameter is entered, the command will print a detailed description of the sensor identified by that particular sensor number. The command parses the SDR of the desired sensor and displays various information about the sensor:

- name
- type
- value
- sensor units
- sensor state
- sensor maximum and minimum values
- threshold values
- outputs assigned to threshold events
- fan control groups the sensor is assigned to (only for temperature sensors)
- active level for discrete sensors
- input type for input sensors
- output type for output sensors
- active drivers for output sensors

Example 2. Get detailed information about the 5V sensor

```
%>sensor 2
------------------Sensor Details-------------------
* Name: V0
* Type: Threshold
* Value: 3.21
* Sensor Units: V
* State: Lower Non-Recoverable
* Sensor Maximum Reading: 5.00
* Sensor Minimum Reading: 0.00
* Upper non-recoverable threshold: 4.12
    Assigned Outputs:  None
* Upper critical threshold: 3.80
    Assigned Outputs:  None
* Upper non-critical threshold: 3.49
    Assigned Outputs:  None
* Lower non-critical threshold: 2.86
    Assigned Outputs:  None
* Lower critical threshold: 2.55
    Assigned Outputs:  None
* Lower non-recoverable threshold: 2.23
    Assigned Outputs:  None
* Assertion Events logged for: unr uc unc lnc lc lnr
* De-Assertion Events logged for: unr uc unc lnc lc lnr
* Positive-going threshold hysteresis value: 0.02
* Negative-going threshold hysteresis value: 0.02
* Debounce: 0
```

The local_sensor command can also be used to change different sensor parameters. Using the *sensor* command the following parameters can be changed:

- temperature sensors : thresholds, hysteresis, fan control active mask, outputs assigned to threshold events
- fan speed sensors: thresholds, hysteresis, outputs assigned to threshold events
- input sensors: active level, outputs assigned to active level,input type
- output sensors: active level, output drivers logic function, input type
- voltage: thresholds, hysteresis, outputs assigned to threshold events
- current: thresholds , hysteresis, output assigned to threshold events

6.6.1 Changing a threshold's value | Disabling a Threshold

Syntax: **local_sensor** sensor_no **threshold** threshold_code *value*| *disable*

- threshold_code is defined in Table 3: Threshold_Code

The sensors used to monitor the system's parameters are described by full sensor records and support up to 6 thresholds. The threshold can be enabled or disabled by this command.
To activate a disabled threshold all you have to do is set an appropriate value for it. To disable an active threshold you have to use this command with the disable parameter.

The *threshold_code* parameter represents an abbreviation of the threshold that needs to be changed.

The value change operation will be successful only if the new value for the threshold is compliant to the monotony rule for thresholds:

lnr < lc < lnc < unc < uc < unr

The new value is compared only with the values of active thresholds.

Example 3. Changing the upper critical threshold of temp 3 (sensor 28)

```
%>local_sensor 28 threshold uc 45
 Operation Successful!
```

Example 4. Disabling the upper critical threshold of temp 3 (sensor 28)

```
%>local_sensor 28 threshold uc disable
 Threshold disabled!
```

6.6.2 Changing the hysteresis

Syntax: **local_sensor** sensor_no **hysteresis** hysteresis_code *value*

- hysteresis code is defined in Table 4: Hysteresis_Code

Hysteresis can be modified only for threshold sensors. Sensors described by full sensor records support 2 hysteresis values.
The *hysteresis_code* parameter represents a abbreviation of the hysteresis that needs to be changed.

Hysteresis values can be changed only for sensors that support hysteresis.
In other cases the following warning message will be displayed:

```
Sensor does not support Hysteresis!
```

Example 4. Changing the positive going hysteresis of 3.3V (sensor 2) a

```
%>local_sensor 2 hysteresis pos 0.04
Operation Successful!
```

6.6.3 Changing the outputs linked to a sensor event

Syntax: **local_sensor sensor_no output** [*threshold_code*] *hex_outputs*

- *threshold_code is defined in* Table 3: Threshold_Code
- *hex_outputs is defined in* Table 5: Hex_Outputs

The outputs assigned to sensor events parameter can only be set individually for each sensor and each event.

Outputs can be linked to thresholds infringements for threshold sensors or to the active level for discrete sensors.

For threshold sensor the command uses the threshold_code parameter for identifying the event to which the outputs defined by *hex_outputs* are linked.

In the case of discrete sensor the outputs are automatically linked with the active level so the t*hreshold_code* parameter is not used.

If a bit of *hex_outputs* is 1 the corresponding output will be linked to the corresponding event. For example if outputs 1, 5 and 8 need to be linked with an event *hex_output* = 0x0091.

Example 5. Assigning outputs 1 and 2 to the lower critical threshold of fan 2(sensor 38)

```
%>local_sensor 38 output lc 0x0003
 Operation Successful!
```

Example 6. Assigning outputs 7 and 14 to the active level of input 6(sensor 69)

```
%>local_sensor 69 output 0x2040
 Operation Successful!
```

6.6.4 Changing the activelevel

Syntax: **local_sensor** sensor_no **activelevel** *0|1*

The activelevel can be changed only for discrete sensors.
The active value can be either 0 (low) or 1 (high).

Example 7. Setting the activelevel to high for input 2 (sensor 65)

```
%>local_sensor 65 activelevel 1
 Operation Successful!
```

6.6.5 Changing the fancontrol mask

Syntax: **local_sensor** sensor_no **fancontrol** *fancontrol_mask*

- *fancontrol_mask defined in* Table 6: Fancontrol_Mask

The fancontrol parameter is used only for temperature sensors and determines if the sensor is active for any of the fan control groups. For more details on fan control refer to 1.4.1 Fan control .

The fancontrol_mask has 8 bits, but only the least significant 3 are used. The fancontrol_mask is inputed as a hex value. A sensor that is active for all the fancontrol groups has a fancontrol_mask of 0x07.

Example 8. Setting temp 3(sensor 28) active for PWM 1 and 3

```
%>local_sensor 28 fancontrol 0x05
 Operation Successful!
```

6.6.6 Changing the output type

Syntax: **local_sensor** sensor_no **output_type** *AND | OR*

The *output_type* parameter is available only for output sensors and determines the logic function the sensor performs on it's driving signals.

An **AND** output is asserted when all it's driving signals are asserted.

An **OR** output is asserted when at least one of it's drivers is asserted.

Example 9. Setting the logic function of output 1 to *AND*

```
%>local_sensor 80 output_type AND
 Operation Successful!
```

6.6.7 Changing the input type

Syntax: **local_sensor** sensor_no **input_type** *mswitch | on_off*

The *input_type* parameter is available only for input sensors and determines the way the value of the sensor, assigned to the digital input, changes.

There are two input types:

- ON – OFF – regular digital input
- Momentary Switch – this type of input is used for push buttons; the sensor assigned to a input of this type has it's value toggled at every active level of the input

Example 10. Setting input 1 as a momentary switch

```
%>local_sensor 64 input_type mswitch
 Operation Successful!
```

6.6.8 Asserting/Deasserting an output

Syntax: **local_sensor** *sensor_no* assert|deassert

This command is used to assert or deassert one of the 16 digital outputs. Only the ***admin*** can assert or deassert outputs. The ***user*** has it's permission denied when he tries to use this command.

Example 11. Asserting Output 4(sensor 83)

```
%>local_sensor 83 deassert
 Done!
```

6.6.9 Changing the user's permission to assert/deassert an output

Syntax: **local_sensor** *sensor_no* **userpermission** allow|deny

By default only the admin can assert/deassert the digital outputs. The user is denied when he tries to change the state of the outputs.

This command is used to allow or deny the user to change a particular output.

Example 11. Allow the user to change output 4(sensor 83)

```
%>local_sensor 83 userpermission allow
 Done!
```

6.6.10 Changing the timeout for an input

Syntax: **local_sensor** *sensor_no* **timeout** value

The timeout option is available only for sensors assigned to digital inputs.

The digital outputs can have more than one driver. If one of the drivers is a digital input, it's timeout value can be used to define a time period in which only the input drives the output. In the timeout interval only the value of the input determines the outputs value,and all the other drivers are ignored. After the timeout passes all the other drivers become active again and the output is controlled according

to it's logic function.

Ex: Input 1 (used as a Power on switch), and Fan 1 lc threshold(fan fail) are controlling Output 1(a shutdown signal). The shutdown signal should be active if either Power on is inactive or if the fans fail. But at start-up when the Power on switch is pressed, the Fans are stopped and the Fan fails signal will also be asserted. In this case even if the Power on is pressed , the shutdown signal is still asserted due to the fans being stopped. If the timeout option is used for the active level of Input 1 (Power on switch), the input becomes the only driver for the output until the timeout passes, thus allowing the fans sufficient time to start and reach a stable speed before they are taken into consideration for shut down.

By default all inputs have a timeout value of 0 (disabled).

Timeout values = 0..15 seconds

Example 11. Set a 2 seconds timeout for input 2(sensor 65)

```
%>local_sensor 65 timeout 2
 Done!
```

6.6.11 Changing the debounce option

Syntax: **local_sensor** sensor_no debounce *debounce_value*

- sensor_no : identifies the sensor that will be set up to drive the led.
- debounce_value: 0-254

Enables or Disables the debounce option for a particular sensor. Debounce is available only for threshold,discrete and input sensors.

The debounce parameter establishes the number of times a sensor is allowed to be out of spec before it is reported as failing. If debounce is 0 the sensor is reported as failing the first time it is out of spec.

Example 17. Set debounce value for sensor 2

```
%>local_sensor 2 debounce 2
 Done!
```

In this case, sensor 2 has to fail 3 times before it is reported as failing.

6.7 fancontrol command

Syntax: fancontrol [sensor**] | [***group_no* **[auto] |**
[manual [level *l_value***]] |**
[minimum level *l_value***] |**
[temp0|temp1|temp2 *t_value***]]**

- group_no : 1 - 3
- manual level and minimum level value : 0 - 15
- t_value: -20..100 Celsius degrees

Functions:

The fancontrol command is used to modify parameters for the fan control algorithm. For more details about the algorithm that is used to control the fan speeds refer to 1.4.1 Fan control.

Each fancontrol group represents a PWM signal. The fan control algorithm uses 15 distinct fan levels to represent the 0-100% duty cycle of the PWM signal. Fan level 0 is equivalent to 0% and fanlevel 15 is equivalent to 100% duty cycle for the PWM signal.

6.7.1 Viewing the fancontrol status

Syntax: fancontrol

If the command is used without any parameter it displays the current state of all the fan control groups and their respective parameters.

Example 1. Viewing the fancontrol status

```
%>fancontrol
-------------------------------------------------------------------------
Fan   Control   Current    Manual     Minimum   Temp0  Temp1  Temp2  Maximum
Group  Method  Fan level  Fan Level  Fan level                        Temp
-------------------------------------------------------------------------
  1    Manual     7          7          3        0      30     60      24
  2    Auto       3          3          3        0      30     60      24
  3    Auto       3          3          3        0      30     60      24
```

The fans can be controlled automatically using a control algorithm or manually by specifying a fan level. By default all fan groups are controlled automatically.

6.7.2 Setting a fangroup for auto control

Syntax: fancontrol *group_no* **auto**

For controlling the fans speed automatically a set of the installed temperature sensors is used. Every fan group has his own distinct set. The fan speed is controlled in accordance with the algorithm described in chapter 1.4.1 Fan control and the maximum temperature of the sensors in it's set.

Example 2. Setting fancontrol group 2 to auto

```
%>fancontrol 2 auto
 Done! Fan Group 2 is controlled automatically!
```

6.7.3 Viewing the temperature sensors associated with each fan group

When in auto mode the fan level is controlled by the fan control algorithm, and the maximum temperature in the temperature sensors set that is used for that particular fan group.

Syntax: fancontrol sensor

Example 3. Viewing the temperature sensors associated with each fan group

```
%>fancontrol sensor
 Sensors assigned to Fan Group 1:
   Sensor 26 Temp1 : 23.00 deg C
   Sensor 30 Temp5 : 24.00 deg C
   Sensor 31 Temp6 : 24.00 deg C
 Sensors assigned to Fan Group 2:
   Sensor 26 Temp1 : 23.00 deg C
   Sensor 30 Temp5 : 24.00 deg C
   Sensor 31 Temp6 : 24.00 deg C
 Sensors assigned to Fan Group 3:
   Sensor 27 Temp2 : 23.00 deg C
   Sensor 28 Temp3 : 23.00 deg C
   Sensor 29 Temp4 : 23.00 deg C
   Sensor 31 Temp6 : 24.00 deg C
```

6.7.4 Setting a fangroup for manual control

Syntax: fancontrol *group_no* **manual**

When a fan control group is controlled manually, it's fan level is specified by the manual level parameter.

Example 4. Setting fancontrol group 1 to manual control

```
%>fancontrol 1 manual
 Done! Fan Group 1 is controlled manually using the manual Fan level value!
```

6.7.5 Changing the manual fancontrol level

Syntax: fancontrol *group_no* **manual level** *level_value*

The value for the manual fan level can be choosen between the 15 allowed levels. For level 0 the fans are stopped (PWM duty cycle 0%) and for level 15 the fan are running at full speed (PWM duty cycle 100%)

Example 5. Changing the manual fan level for fan group 3

```
%>fancontrol 3 manual level 13
 Manual Fanlevel = 13
```

Using the fancontrol command the user can change the value for all the fan control algorithm's parameters : minimum level, temp0, temp1, temp2. Each fan group has it's own parameters.

6.7.6 Changing the minimum level parameter of the fancontrol algorithm

Syntax: fancontrol *group_no* **minimum level** *level_value*

The minimum level parameter determines the speed of the fans when the maximum temperature of the sensors is bellow the temp0 parameter.

Example 6. Changing the minimum fan level for fan group 1

```
%>fancontrol 1 minimum level 1
 Minimum Fanlevel = 1
```

6.7.7 Changing the temp0,temp1,temp2 parameters of the fancontrol algorithm

Syntax: fancontrol *group_no* **temp0|temp1|temp2** *temp_value*

The **temp**x parameters are used by the fancontrol algorithm. For more details refer to 1.4.1 Fan control.

Example 7. Changing the temp1 parameter for fan group 1

```
%>fancontrol 1 temp1 34

 Temp1=34
```

6.8 pwm command

Syntax: pwm [*pwm_no* **freq** *freq_value***]**

- pwm_no : 1 - 3
- freq_value : 1 – 125 Khz

Functions:

If used without parameters displays the status of all the PWM signals.
The command can also be used to change the frequency of one of the 3 PWM signals.

Example 1. Status reading for pwm signals

```
%>pwm

 Pwm1 Freq: 25 Khz, Duty Cycle:28%
 Pwm2 Freq: 25 Khz, Duty Cycle:100%
 Pwm3 Freq: 25 Khz, Duty Cycle:100%

*Pwm1 and Pwm2 share the same frequency, but can have different duty cycles
```

The pwm command can also be used to change the frequency of the PWM signals.

! PWM1 and PWM2 share the same frequency.
If the frequency is changed for one signal it will change for both.
Even tough the frequency is the same for the two PWM signals the duty cycles are independent .

Example 2. Changing PWM1 frequencies to 25 KHz

```
%>pwm 1 freq 25
 Done!
 *Pwm1 and Pwm2 share the same frequency, but can have different duty cycles.
```

6.9 xmodem command

Syntax: xmodem configfile | localfru | shelffru | shmcfru | web

Functions:

Sends via RS232 a configuration file, a fru file for a local FRU, the fru file for the Shelf, the fru file for the ShMC or the web page. After the command is entered, the ShMC goes into data receive mode and waits for the data to be sent . You can then start the file transfer with your terminal program and select XMODEM as the protocol.

When using "**xmodem**" in "Hyperterminal" the transfer of the desired file can take up to 10 seconds to start.
Only the admin can use the xmodem command.

Example 1: Sending of the config file

```
%>xmodem configfile
 Please upload the file...
%>...Done!
```

6.10 logout command

Syntax: logout

Function:
Logs out the current user and permits a new log in.

6.11 scispeed command

Syntax: scispeed 9600 | 19200 | 38400 |115200

Functions:
Changes the baud rate at which the CLI for the ShMC and the bootloader framework operate. For the change to become valid the environment has to be saved using the saveenv command and the ShMC has to be restarted, either using the reboot command or by using the reset key. The baud rate is changed at the next power up.

Example 1 :

```
 %>scispeed 9600
Baud rate changed to 9600.Save Environment and reboot.
```

6.12 passw command

Syntax: passw [snmp auth | priv]

Function:
Changes the log-in or snmp passwords for the current user. For log-in password, if instead of the new password an empty string is entered, password checking for the respective user is disabled.

6.13 reboot command

Syntax: reboot

Function:

Restarts the ShMC.

6.14 uptime command

Syntax: uptime

Function:

Displays the amount of time which has past since the last ShMC power up.

6.15 saveenv command

Syntax: saveenv

Function:

Saves the parameters that were changed. If the modified parameters are not saved, they will be lost at the next reboot.

6.16 version command

Syntax: version

Function:

Displays information about the ShMC : Part Number,Software Version,MAC Address, and Serial Number.

6.17 restore command

Syntax: restore

Function:

Restores all parameters to the default values. For the restore to be complete a reboot is necessary.

6.18 lanconfig - command

Syntax:

Syntax:

lanconfig [settings] | [ip | mask| gateway [*address***]] | [dhcp [hostname | clientid] [on | off]]]**

Functions:

Readout or setting of network parameters.

- no parameters – return of IP, mask and gateway addresses of the LAN interface
- **settings** – interface settings: dhcp or local control for addresses
- **ip** – returns IP address
- **mask** – returns network mask
- **gateway** – returns standard gateway
- address – if any address is inputed, it's value is assigned to the parameter entered before it
- **dhcp –** displays the current dhcp state -enabled/disabled and dhcp configuration options.
- **hostname** – if turned on will enable the DHCP option 12 hostname.
- **clientid** – if turned on will enable the DHCP option 61 client id. The format of the

client ID option follows the requirements form PICMG HPM.3 R1.0 specification

- **on** – if entered after **dhcp** enables it
- **off** – if entered after **dhcp** disables it

After a new address is set, for the change to become effective, it must be saved with **saveenv** and the ShMC restarted, either using the reboot command or using a power cycle.

Example 1: Readout of the IP address

```
%>lanconfig ip

IP=193.155.166.51
```

Example 2: Changing the IP address

```
%>lanconfig ip 196.100.100.1

IP=196.100.100.1
```

Example 3: Enable dhcp

```
%>lanconfig dhcp on
 DHCP on
```

6.19 sol command

Syntax: sol [on|off]

Functions:

Displays or changes the current status of the Serial Over Lan (SOL) configuration. Whenever the SOL mode is activated the behavior of RS232 port will change. All the data received on the serial port will be transmitted in telnet packets over Ethernet to all telnet connections opened on "serial" profile. The data received from the telnet connections opened in "serial" mode is transferred to the serial port.

6.20 date command

Syntax: date[dd.mm.yyyy]

Functions: displays or sets the current date

Example 1: Check out of date

```
%>date

 Date [dd.mm.yyyy] = 04.09.2012
```

Example 2: Setup of date

```
%>date 13.10.2013
 Done!
```

6.21 time command

Syntax: time[hh:mm:ss]

Functions: displays or sets the current time

Example 1: Check out of time

```
%>time

 Time [hh:mm:ss] = 16:52:27
```

Example 2: Setup of time

```
%>time 16:53:00
 Done!
```

6.22 sel command

Syntax: sel count | clr | print [*start_record_no* **[***end_record_no***]]**

Functions:

The command can display the number of sel entries, display a particular set of these entries or clear the sel.

To print the whole log use : ***sel print***

To print all the records starting with a particular one use : ***sel print*** *record_no*

To print all records in a give interval: ***sel print*** *start_record_no* *end_record_no*

Example 1: Readout of SEL

```
 %>sel print
------------------------------------------------------------------------------
                             Sensor  Event Log

Rec.ID. dd.mm.yyyy hh:mm:ss  Sensor No. Name             Event  Ev.Dir  Value  Threshold
------------------------------------------------------------------------------
------------------------------------------------------------------------------
0x0001  01.01.2012 00:00:00     97      SHMC Power ON     1 (Asserted)
0x0002  01.01.2012 00:00:00     2       V0                LNC     As      0.00     2.86
0x0003  01.01.2012 00:00:00     4       V2                LC      As      0.00     11.38
0x0008  01.01.2012 00:00:00     7       V5                LC      As      0.00     0.66
0x0009  01.01.2012 00:00:00     9       V7                LC      As      0.00     12.39
0x000A  01.01.2012 00:00:00     2       V0                LC      As      0.00     2.55
0x000C  01.01.2012 00:00:00     64      Input1            1 (Asserted)
0x000D  01.01.2012 00:00:00     65      Input2            1 (Asserted)
0x000E  01.01.2012 00:00:00     66      Input3            1 (Asserted)
0x000F  01.01.2012 00:00:00     67      Input4            1 (Asserted)
0x0019  01.01.2012 00:00:00     77      Input14           1 (Asserted)
0x001A  01.01.2012 00:00:00     78      Input15           1 (Asserted)
0x001B  01.01.2012 00:00:00     79      Input16           1 (Asserted)
```

Example 2: SEL clearing

```
 %>sel clr
  Done! Sel is empty!
```

6.23 help command

Syntax: help [-v]

Functions:

Displays a list of all available commands or a more detailed description if the verbose attribute is used (-v).

6.24 snmp command

Syntax: snmp [user | admin [auth [md5 | none] | privacy [des | aes | none]]]

Functions:

Configure the SNMP v3 authentication and privacy settings for user and admin profiles.

6.25 eth command

Syntax: eth [telnet | web [on | off]]

Functions:

Enables/disables the telnet and web interfaces

6.26 tftp command

Syntax: tftp firmware | web | config *tftp_server_ip_address filename*

Functions:

Uploads new firmware, web page or configuration file by TFTP.

6.27 shelfaddr command

Syntax: shelfaddr [set [-cpci] <value>]

Functions:

Displays or changes the Shelf Address. If the "-cpci" options is used the address is a two bytes value as defined in PICMG 2.9. If the "-cpci" options is not used the address is a maximum 20 characters text, as defined in PICMG 3.0. The shelf address field is used to uniquely identify a chassis when the DHCP clientID option is used.
The PICMG 3.0 defined address is stored into the Address table record, if this record is present in the Shelf FRU Information. Whenever the Address table record is present into the Shelf FRU Information, the Shelf Address field of this record is used. If this field is empty or the record is not present, the PICMG 2.9 address is used for CPCI systems only.

6.28 hostname command

Syntax: hostname [<name>]

Functions:

Displays or changes the Hostname of the Shelf Manager

7 Restore to factory defaults procedure

Only an admin can perform a system restore. In order to restore all parameters to their default value the following steps need to be followed:

- Login using the admin account (for more details refer to 5. Command Line Interface (CLI))
- Use the restore command

! When using restore the ShMC disregards all the changes applied to the SDRs, Sensor options and User Settings and uses a predetermined set for all these parameters. This predetermined set of parameters may be different then the one loaded on the ShMC when it was shipped out.

To go back to a particular setup you have to use xmodem configfile, and upload a Configuration file that contains the required setup.